AMENDMENTS TO THE SPECIFICATION

Please amend Paragraph [0022] as follows:

[0022] The preferred embodiment starts by placing the new authenticator in position, but keeps the old authenticator up and running to help process entries where needed. When a user attempts to login, the method uses servlets to first check to see if the user's password exists within the datastore for the new authenticator. If not, then the user is sent to the old authenticator's sign-on page. When the user enters their authentication credentials into the old authenticator, a second servlet will capture this information and wait to see if the user is properly authenticated (to make sure the credentials are correct). As a preferred alternative, where possible, a code snippet attached to the login process could accomplish the function of this second servlet. The capture may be made during entry of the authentication credentials into the old authenticator or may occur after confirmation of proper authentication. Regardless, once the user is confirmed by the old authenticator, the information captured will be used to populate the new datastore, and the user will be redirected (by HTTP) to the proper page they were requesting. The user may or may not be asked to reauthenticate by the new **~~authentideator~~** **authenticator** as a final step before redirection. Of course, the next time the user accesses a page and needs to be authenticated, the first servlet will note that a password is already present in the new datastore and automatically forward them to the new authenticator using the new datastore for authentication.

Please amend Paragraph [0028] as follows:

[0028] In action 1A, the non-encrypted data is migrated from the first authenticator's datastore 60 (in this example an RDBMS) to the second authenticator's datastore 65 (in this example an LDAP-compliant directory service). This will include the roles, resources, and user information (except password). The passwords can not be migrated in this fashion, since they are stored in a proprietary encryption algorithm. During this migration the roles and resources can also be translated to comply with new naming standards or with appropriate syntax for the second ~~**authenticator.In**~~ **authenticator. In** action 2A, the second authenticator is installed, and the protected resources 30 moved to a web server working in cooperation with the new authenticator (specifically with its web agent (or plug-in) 25). The first authenticator is taken out of any active role in protecting any web resources and the second authenticator is configured to perform this function.

Please amend Paragraph [0037] as follows:

[0037] Figure 3 illustrates the process for a user who has already successfully completed migration. The user through browser 10 seeks access to one of the protected resources 30 and is intercepted in action 1B by servlet 22. Servlet 22 seeks and obtains identification from the user and in action 2B queries datastore 65 to determine if a password has been populated for that identification. Since the user has already migrated, there will be a confirmation that the password is present and the browser will be directed to web agent 25. In action 3B web agent 25 will provide its login page for the user to complete. In a preferred embodiment, servlet 22 will prepopulate the login page with the identification it already possesses to save the user a step. In an alternative embodiment, servlet 22 could seek both identification and password, check identification, and then pass both on (by prepopulating the login screen or by direct submission through a different hook or pathway) to whichever authenticator is to be used for authentication on that go around by the user (which would apply to the description of Figure 2 as well). In action 4B web agent 25 provides the authentication information (identification and password) to policy server 55 which verifies the information in action 5B from datastore 65. In action 6B, policy server 55 provides confirmation of authentication and typically authorization information as ~~**wellto**~~ **well to** web agent 25. Web agent 25, in action 7B, then may provide a cookie to browser 10 and provides access to the requested resource 30.